A Generalized Frequency Weighting Framework for LQG Compensator Design

Michael Sherback

Sibley School of Mechanical and Aerospace Engineering
Cornell University, Ithaca NY 14853 USA
mas61@cornell.edu

Summary. This paper specifies methods enabling specific types of frequency domain loopshaping in the LQG framework. It gives the augmented state and observation equations for a general system with colored sensor or motor noise, sensor and actuator dynamics, and frequency weight on the control and performance costs. The performance weights are useful in design for dealing with the effects of many challenges facing control designers other than colored control and performance in the narrow sense, through shaping the sensitivity and loop gain. Delay is useful both as a common defect in applications, and as a tool to investigate robustness. An example of using the associated Matlab routine is given. This is nothing earth shattering but it is useful but tedious work that readers won't need to repeat.

1 Introduction

This paper is describes the implementation of a very general way to use frequency weighted performance in LQG. It gives the equations for that along with a somewhat obvious but still tedious framework for modelling the system's sensor and actuator dynamics. These are not huge conceptual leaps, just the working-out of a convenient and general framework.

The main contribution is just writing the equations out in matrix form and making the whole thing available in Matlab. You can get the key files by emailing me.

2 Equations of the augmented system

We seek to design a feedback controller for a plant described in standard state space form: $\dot{x}_g = A_g x + B_{gu} u + B_{gw} w$, and $y = C_g x + D_{gu} u + D_{gw} w$. In order to design a controller with sensor and actuator dynamics, and frequency weighting on performance, a series of additional plants are defined based on the convention

$$\dot{x_x} = A_x + B_x u_x, y = C_x x_x + D_x u_x \tag{1}$$

- Subscript z denoting performance and input frequency weighting.
- Subscript u denoting actuator dynamics.
- Subscript y denoting sensor dynamics.
- Subscript w denoting disturbance coloring.
- Subscript v denoting sensor noise coloring.
- Subscript k denoting Kalman filter dynamics.

Each of these plants takes as an input a variable with the subscript i and outputs a variable whose name is the same as the subscript of the system, i.e. $z = G_z z_i$. The internal state of the performance system is x_z and so on for the rest. The content of u_i, y_i, w_i, v_i should be obvious. The input to the performance subsystem $z_i = [x_g; u]$, but it is most convenient to express the system equations without using z_i. Note that control inputs are fed through to the performance subsystem to allow for frequency weighting.

It is useful to have some special B and D matrices that don't fit the above form and have dual subscripts, in order to make equations readable: $B_{zu}, D_{zu}, B_{zg}, D_{zg}$ are partitions of B_z, D_z, used to feed the plant state and actuator output to the performance. It would not be too hard to allow for feeding the internal actuator states or even sensor output or states to the performance if desired but it seems excessive now. The plant takes inputs from multiple subsystems so for clarity its input matrices are partitioned into $B_{gu}, D_{gu}, B_{gw}, D_{gw}$.

The closed loop system is shown in fig. 1, with the state feedback gain K. Apologies if it is insultingly explicit, but I wanted to be thorough. For the purposes of this paper, the compensator is considered to include everything from y_i to u_t. The disturbance vector w may include exogenous or endogenous noises or references.

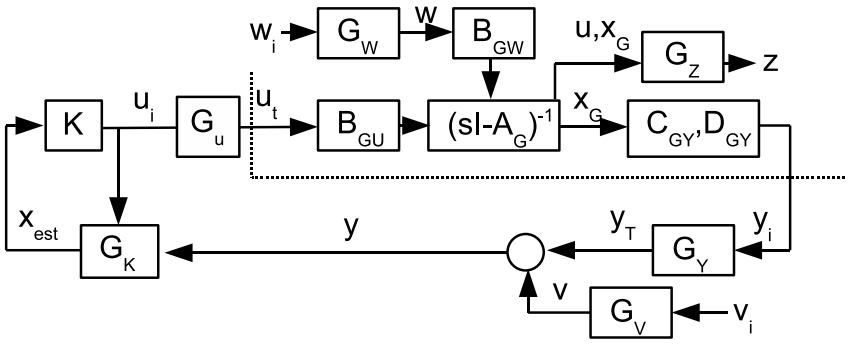

Fig. 1. System Loop.

The nicest thing about this format and how it is implemented in Matlab is that you can just set any of these systems to be feedthrough. By setting A and B to empty matrices, C to zero, and D to identity, and allowing the multiplication of a matrix by an empty matrix to yield an empty matrix, the equations simplify to exactly what you would get had you not done the more general form, and correspond to fig. 2.

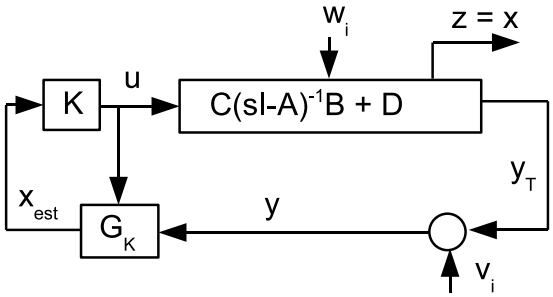

Fig. 2. System Loop with all feedthroughs.

The governing equations follow. Apologies for tedium but I just wanted to make it complete. First, the disturbance and performance coloring along with actuator dynamics are described by eq. 1 with no double subscripts.

Here are the plant equations to y_i, a noiseless pre-sensor dynamics variable.

$$\dot{x}_g = A_g x_g + B_{gu} u + B_{gw} w \tag{2}$$
$$y_i = C_g x_g + D_{gu} u + D_{gw} w \tag{3}$$

For making them into matrix equations it is useful to expand them

$$\dot{x}_g = A_g x_g + B_{gu} C_u x_u + B_{gu} D_u u_i + B_{gw} C_w x_w + B_{gw} D_w w_i \tag{4}$$
$$y_i = C_g x_g + D_{gu} C_u x_u + D_{gu} D_u u_i + D_{gw} C_w x_w + D_{gw} D_w w_i \tag{5}$$

Here are the equations for the performance weighting system.

$$\dot{x}_z = A_z x_z + B_{zg} x_g + B_{zu} u \tag{6}$$
$$z = C_z x_z + D_{zg} x_g + D_{zu} u \tag{7}$$

or, expanded,

$$\dot{x}_z = A_z x_z + B_{zg} x_g + B_{zu} C_u x_u + B_{zu} D_u u_i \tag{8}$$
$$z = C_z x_z + D_{zg} x_g + D_{zu} C_u x_u + D_{zu} D_u u_i \tag{9}$$
$$\tag{10}$$

Sensor equations follow and then their expanded forms. The subscript t denotes 'true', as the output of the sensor dynamics is not yet corrupted by noise.

$$\dot{x}_y = A_y x_y + B_y y_i \tag{11}$$
$$y_t = C_y x_y + D_y y_i \tag{12}$$
$$\dot{x}_y = A_y x_y + B_y C_g x_g + B_y D_{gu} C_u x_u + B_y D_{gu} D_u u_i ... \tag{13}$$
$$+ B_y D_{gw} C_w x_w + B_y D_{gw} D_w w_i \tag{14}$$
$$y_t = C_y x_y + D_y C_g x_g + D_y D_{gu} C_u x_u + D_y D_{gu} D_u u_i ... \tag{15}$$
$$+ D_y D_{gw} C_w x_w + D_y D_{gw} D_w w_i \tag{16}$$

The complete open loop system matrix equation does not fit on one line. It is:

$$\left[\dot{x}_1\right] = \begin{bmatrix} \dot{x}_z \\ \dot{x}_y \\ \dot{x}_g \\ \dot{x}_u \\ \dot{x}_w \\ \dot{x}_v \end{bmatrix} = \left[A_1\right] \begin{bmatrix} x_z \\ x_y \\ x_g \\ x_u \\ x_w \\ x_v \end{bmatrix} + \left[B_1\right] \begin{bmatrix} u_i \\ w_i \\ v_i \end{bmatrix} = \left[B_1\right]\left[u_1\right] \tag{17}$$

where

$$\left[A_1\right] = \begin{bmatrix} A_z & 0 & B_{zg} & B_{zu}C_u & 0 & 0 \\ 0 & A_y & B_y C_g & B_y D_{gu} C_u & B_y D_{gw} C_w & 0 \\ 0 & 0 & A_x & B_{gu} C_u & B_{gw} C_w & 0 \\ 0 & 0 & 0 & A_u & 0 & 0 \\ 0 & 0 & 0 & 0 & A_w & 0 \\ 0 & 0 & 0 & 0 & 0 & A_z \end{bmatrix} \tag{18}$$

and

$$\left[B_1\right] = \begin{bmatrix} B_{zu} D_u & 0 & 0 \\ B_{gu} D_u & B_{gw} D_w & 0 \\ B_u & 0 & 0 \\ 0 & B_w & 0 \\ 0 & 0 & B_v \end{bmatrix} \tag{19}$$

and the outputs are given by

$$\begin{bmatrix} y_t \\ y \\ z \end{bmatrix} = \begin{bmatrix} C_{1yt} \\ C_{1y} \\ C_{1z} \end{bmatrix} \left[x_1\right] + \begin{bmatrix} D_{1yt} \\ D_{1y} \\ D_{1z} \end{bmatrix} \left[u_1\right] \tag{20}$$

where

$$\begin{bmatrix} C_{1yt} \\ C_{1y} \\ C_{1z} \end{bmatrix} = \begin{bmatrix} 0 & C_y & D_y C_x & D_y D_{gu} C_u & D_y D_{gw} C_w & 0 \\ 0 & C_y & D_y C_x & D_y D_{gu} C_u & D_y D_{gw} C_w & C_v \\ C_z & 0 & D_{zg} & D_{zu} C_u & 0 & 0 \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} D_{1yt} \\ D_{1y} \\ D_{1z} \end{bmatrix} = \begin{bmatrix} D_{gu} & D_u D_{gw} & 0 \\ D_{gu} & D_u D_{gw} & D_v \\ D_{zu} D_u & 0 & 0 \end{bmatrix} \tag{22}$$

This can be expressed in the standard linear fractional transformation format as

$$\begin{bmatrix} A_1 & [B_{1w} \; B1v] & B_{1u} \\ C_{1z} & 0 & D_{zu} D_u \\ C_{1yt} & [D_{gw} \; D_w] & D_{gu} D_u \end{bmatrix} \tag{23}$$

The next step is to do LQR on this. There is a little bit of trickery required to handle passing the control input through to the performance, similar to something done in MAE 678 on March 2, 2005. The z vector corresponding to the R_{zz} matrix in this formulation will generally be first some frequency weighted function of the plant state, followed by a frequency weighted actuator output. It helps to define a partitioning of $D_{1z} : D_{2z} = D_{zu} D_u$, and then the relevant equations are

$$z = C_{1z} x_1 + D_{2z} u_i \tag{24}$$

$$J = \int [z^T R_{zz} z + u_i^T R u_i] dt \tag{25}$$

$$= \int [(C_{1z} x_1 + D_{2z} u_i)^T R_{zz} (C_{1z} x_1 + D_{2z} u_i) + u_i^T R u_i] dt \tag{26}$$

$$= \int [x_1^T C_{1z}^T R_{zz} C_{1z} x_1 + 2 x_1^T C_{1z}^T R_{zz} D_{2z} u_i + u_i^T (D_{2z}^T R_{zz} D_{2z} + R) u_i^T] dt \tag{27}$$

This is handled in a standard way in Matlab's lqry.m.

The Kalman Filter is obtained in the standard way with kalman.m, using noise intensities from before the prewhiteners, and yields a subsystem G_k.

The final step is to obtain the equations for the stand-alone compensator. It is easy if you consider the actuator and sensor to be outside of the compensator. If you want to include them, then it is helpful to partition the Kalman filter B_k matrix into B_{ku} and B_{ky} corresponding to the different inputs to the filter, and then the equations are:

$$\dot{x}_p = \begin{bmatrix} A_y & 0 & 0 & 0 \\ 0 & Av & 0 & 0 \\ B_{ky} C_y & B_{ky} C_v & A_k - B_{ku} K & 0 \\ 0 & 0 & -B_{ku} K & A_u \end{bmatrix} \begin{bmatrix} x_{py} \\ x_{pv} \\ \hat{x} \\ x_u \end{bmatrix} + \begin{bmatrix} B_y & 0 \\ 0 & B_v \\ B_{ky} D_y & B_{ky} D_v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y_t \\ v_i \end{bmatrix} \tag{28}$$

$$\begin{bmatrix} u_t \end{bmatrix} = \begin{bmatrix} 0 \; 0 \; -D_u K \; C_u \end{bmatrix} \begin{bmatrix} x_{py} \\ x_{pv} \\ \hat{x} \\ x_u \end{bmatrix} \tag{29}$$

It is also helpful to have the closed loop system equations from v and w to u_i. Expressing the state as a stacked vector of the true and estimated x_1, the A and B matrices are

$$\begin{bmatrix} \dot{x_1} \\ \dot{\hat{x_1}} \end{bmatrix} = \begin{bmatrix} A_1 & -B_{u1}K \\ LC_{1y} & A_1 - B_{u1}K - LC_{1y} \end{bmatrix} \begin{bmatrix} x_1 \\ \hat{x_1} \end{bmatrix} + \begin{bmatrix} B_{w1} & B_{v1} \\ 0 & LD_v \end{bmatrix} \begin{bmatrix} w_i v_i \end{bmatrix} \quad (30)$$

The C and D matrices for y are $[C1y0], [D1y]$. The C and D matrices for y_i are $[0_z 0_y C_g D_{gu} C_u D_{gw} C_w - D_{gu} D_u K], [D_{gw} D_w 0]$. Knowing that $u_i = -K\hat{x_1}$ then C and D matrices to u_i are $[0 - K], 0$.

For the purposes of predicting noisy actuator output, it is useful to have a system with inputs w, v and output u, but endogenous noise is then both an input and an output. This is dealt with by requiring $G_u = G_m$ (often reasonable) with subscript m denoting the endogenous disturbances (actuator noise) subset of w. Using superscript m to denote a matrix modification in which the terms for non-endogenous disturbances are set to zero, defining an output $u_{ui+mi} = -K\hat{x} + C_w^m x_w^m + D_w^m + w_i^m$ so that $u = G_u u_{ui+mi}$ it is possible to simulate the closed loop system to the actuator's input, and then post-filter this with $G_u = G_m$ to obtain u. The output is obtained with $C_{CL,ui+mi} = [0_p 0_y 0_g C_w^m 0_v - K], D_{CL,ui+mi} = [D_w^m 0]$.

In general be careful to remember that prewhiteners are built into the system equations if you want to simulate them with lsim or something like that.

3 Utility

The use of sensor and disturbance noise coloring is well known. Sensor and actuator dynamics can be used for obvious purposes, with delay and lag being common.

The more interesting part is the use of frequency weighting on performance and actuation. This can be used to allocate different actuators that control the same degree of freedom to different frequency ranges. For example, a piezo actuator and a DC gearmotor might both be used on a single assembly in order to achieve both fine, high-bandwidth control, and large dynamic range. By penalizing low frequency use of the piezo and vice versa, you can have cheap control of the piezo without saturating it. This can also be used for loopshaping, for example, LQG often makes bad decisions about the allocation of loop gain for highly resonant structures. By putting a band pass filter with some feedthrough on the control cost, loop gain and hence sensitivity spikes in that frequency range can be suppressed. In general, LQG has robustness problems, and so additional loopshaping with these techniques is a good idea in cases were going to a robust control design does not make sense. In many cases it makes sense to penalize high frequency inputs, for

example when actuators have multiplicative noise, as in the human case. In vibration isolation, it is typically necessary to tolerate accelerations at very low and very high frequencies for hardware reasons, and this can be reflected in a state performance weight. Power consumption may be a function of the frequency of actuator input due to inefficiencies. Obviously people often want to suppress high frequency chatter on states with backlash. In vehicle systems, frequency allocation of disturbances is the well known NVH problem.

With care you can use coupling between inputs or states to achieve odd things... I can't think of what to do with that now, but...

4 Conclusion

Enjoy. It is easiest to start with very simple things with feedthroughs everywhere and work your way up.

This work was supported by an NSF Graduate Student Fellowship.

References

1. Kleinman, D. A Control Theoretic Approach to Manned-Vehicle Systems Analysis. IEEE Trans. Automatic Control, Vol. AC-16, no. 6, December 1971.
2. Fagergren A, Ekeberg O, Forssberg H. Precision Grip Force Dynamics: A System Identification Approach. IEEE Trans. Biomedical Engineering, Vol. 47, no. 10, October 2000.